PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04K 1/00, H04L 9/00 — A1

(11) International Publication Number: WO 98/51029

(43) International Publication Date: 12 November 1998 (12.11.98)

(21) International Application Number: PCT/US98/09206

(22) International Filing Date: 7 May 1998 (07.05.98)

(30) Priority Data:
08/852,405 7 May 1997 (07.05.97) US

(71) Applicant: SOUTHWESTERN BELL TELEPHONE COMPANY [US/US]; One Bell Center, St. Louis, MO 63101 (US).

(72) Inventor: JOST, Larry, T.; 10911 Cosmos Drive, St. Louis, MO 63123 (US).

(74) Agent: BERNSTEIN, Bruce, H.; Greenblum & Bernstein, P.L.C., 1941 Roland Clarke Place, Reston, VA 20191 (US).

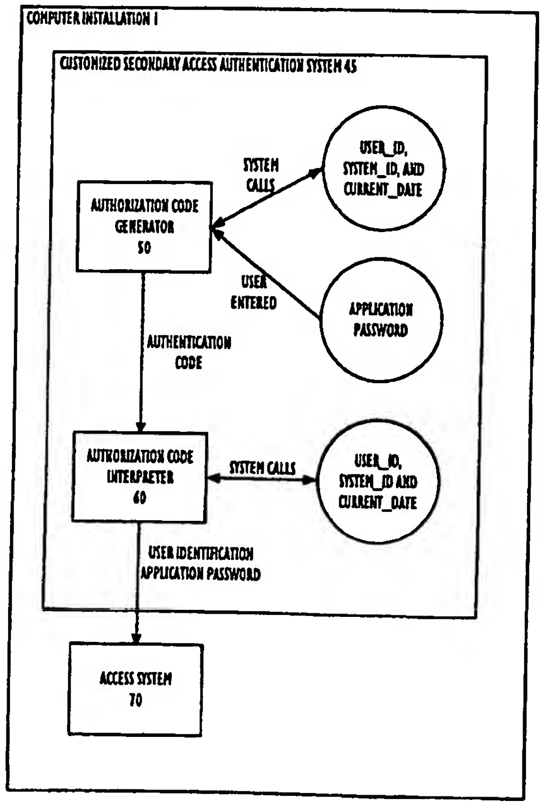
(81) Designated States: JP, KR, MX, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: APPARATUS AND METHOD FOR CUSTOMIZED SECONDARY ACCESS AUTHENTICATION

(57) Abstract

An apparatus and method are provided for secondary access authentication (45) of users who have been primarily authenticated by an operating system level authentication process. An authentication code is utilized that is secure while allowing users to publicly disclose the authentication code. Embedded within the authentication code may be restrictions as to specific host systems, dates and times of access, expiration information, and a key to ensure that changes are made at specified intervals. Further information may be embedded into the AC as such information is deemed necessary. The system also provides for automatic expiration of an application level password and/or application access in accordance with predetermined parameters. The authentication code may be used in conjunction with operating-system level authenticated user identification to validate the authentication code information without exposing the information to others.


COMPUTER INSTALLATION 1
CUSTOMIZED SECONDARY ACCESS AUTHENTICATION SYSTEM 45
SYSTEM CALLS
USER_ID, SYSTEM_ID, AND CURRENT_DATE
AUTHORIZATION CODE GENERATOR 50
USER ENTERED
APPLICATION PASSWORD
AUTHENTICATION CODE
AUTHORIZATION CODE INTERPRETER 60
SYSTEM CALLS
USER_ID, SYSTEM_ID AND CURRENT_DATE
USER IDENTIFICATION APPLICATION PASSWORD
ACCESS SYSTEM 70

TITLE OF THE INVENTION

APPARATUS AND METHOD FOR CUSTOMIZED SECONDARY ACCESS AUTHENTICATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to the field of computer access authentication. More particularly, the present invention relates to secondary-level access authentication performed by application software to authenticate a user who has already been authenticated at an operating system-level and granted primary-level access to a computer installation or host system.

2. Acronyms

The written description provided herein contains acronyms which refer to various telecommunications services, components and techniques, as well as features relating to the present invention. Although some of these acronyms are known, use of these acronyms is not strictly standardized in the art. For purposes of the written description herein, acronyms will be defined as follows:

Application Programming Interface (API)
Authentication Code (AC)
Internet Protocol (IP)
File Transfer Protocol (FTP)
Graphical User Interface (GUI)
Local Area Network (LAN)
Transmission Control Protocol/Internet Protocol (TCP/IP)
Wide Area Network (WAN)

3. Background and Relevant Information

Generally, when a user logs onto a computer installation or host system, the user enters a user identification and a password. The entered user identification and password are compared by the computer installation or host operating system against a password list, password file or database table to determine if the user entered a valid user identification/password combination. If the entered combination is valid, the user is granted access to the computer installation or host system. The above-described validation performed by the host operating system will be generally referred to herein as an "operating system-level authentication" or "primary access authentication".

Once authenticated, a user profile previously defined or established by a system administrator determines the user's privileges on the computer installation or host system. The user profile generally includes the user's permissions (e.g., read, write and execute permissions) at various predefined levels within the directory or folder structure of the computer. For example, UNIX operating systems define an other level, a group level, and a owner level. In UNIX systems, other level permissions apply to all users within the computer installation. Thus, users and system administrators who desire to permit universal user access to particular files and directories/folders may do so by setting the appropriate other level permissions. Group level permissions apply only to users who are members of a predefined group. The group level permissions do not apply to users who are not members of the group. By using group level permissions, a predefined subset of all users may be granted access to read, write and execute files and to traverse directories/folders. Owner level permissions are specific to a particular user, and apply only to the particular user. Generally a user is granted read, write and execute permissions to his or her own files and directories. However, in some circumstances a user may wish to prevent accidental updates of a file by setting the file or directory/folder to read-only (i.e., no write access) at the owner level. Thus, in the above-described computer installation or host system having other, group and owner level permissions, a system administrator may define levels of access that range from almost universal system access (e.g., a development manager requiring access to many folders/directories) to access limited to specific directories/folders and/or files (e.g., a programmer working on specific program modules).

In addition to operating system-level authentication or primary access authentication, many application software packages utilize "application-level authentication" or "secondary access authentication" to grant users access to specific applications (e.g., database and spreadsheet applications) residing on the computer installation or host system. Secondary access authentication occurs only after the user has been authenticated by the primary access authentication mechanism, which is generally performed at the operating system level. Thus, this paradigm operates on the assumption that trusted users (i.e., those who have passed the primary access authentication) are attempting to access and use such applications. Application-level access or secondary access authentication is a concern to system administrators, as an increasingly larger amount of sensitive data is migrated from file cabinets, ledger books and stand-alone personal computers to, for example, network-based database tables and spreadsheets.

Presently, secondary access authentication mechanisms and techniques operate much like primary access authentication, and generally require a user to enter a user identification and application password when accessing an application. A disadvantage of this type of secondary access authentication is that the user is prompted for his or her user identification and application password each time the user accesses the application, which may be several times throughout a day. Another technique for performing secondary access authentication is to use scripts to capture fields containing the user identification and application password via a batch-type script. Scripting has a disadvantage in that the user identification and application password may be exposed for viewing by other users on the system, creating a security risk. In both of the above-described secondary access authentication techniques, a system administration maintains a separate access list of user identifications, application passwords, and possibly expiration data for each particular application running on the system in order to secure access and also to ensure proper aging (i.e., determining when an application password should expire) of the current password. Password aging requires additional software coding and tracking to ensure that the user's application-level access and/or application password expires at the appropriate time and date (as primary password aging is performed at the operating system level). Password aging is a particular problem for temporary or transient users who are intended or designated to use an application for only a short period of time. Further, present techniques of secondary access authentication do not prevent users who have been granted primary access from attempting to gain access to the application by, for example, entering multiple user identification and application passwords in an attempt to gain unauthorized access to the application to view and manipulate sensitive data. In addition, present techniques of secondary access do not provide the capability to quickly and easily include additional user or system information with the user identification and application password to enhance security and limit access to applications.

While prior techniques may have adequately provided secondary access authentication capabilities for smaller installations or installations having relatively few users or special access control needs, prior techniques have not addressed secondary access authentication for larger installations having a large number of users and situations for secure gatch scripting of authentications. In particular, prior techniques have not effectively provided secondary access authentication for a limited period of time without the need for additional software coding to perform password aging to handle different access scenarios, for example, when a user from site A is visiting an installation at site B for a period of a few days or weeks. Prior systems have also not provided a mechanism by which a user can securely embed his or her user identification and application password within a script without the user identification and application password being accessible to others on the system. Further, prior secondary access authorization techniques have not effectively prevented authorized system users from attempting to gain access to restricted applications by repeatedly entering user identification/application password combinations in attempt to gain unauthorized access to an application. In addition, prior secondary access authorization techniques have not provided a flexible mechanism to include additional user and system information with the user identification and application password.

Such features would be highly desirable to system administrators and users who desire a secure secondary access authentication mechanism for application software residing and executed on a computer installation. Such features would also be desirable to system administrators and users of computer installations having transient users who desire to have application passwords and/or access to automatically expire without the need for additional data and coding.

OBJECTS AND SUMMARY OF THE PRESENT INVENTION

In view of the above, the present invention, through one or more of its various aspects and/or embodiments, is presented to accomplish one or more objectives and advantages, such as those noted below.

A general object of the present invention is to provide an apparatus and method for secure secondary access authentication to users of computer-based systems and installations.

Further, an object of the invention is to provide an apparatus and method that provide secondary access authentication for a limited period of time without the need for additional data retention to perform aging.

Another object of the invention is to provided a mechanism by which a user can securely embed his or her user identification and application password within a script without the user identification and application password being accessible to others on the system.

Yet another object of the invention is to provide an apparatus and method for effectively preventing authorized users from attempting to gain access to the applications by repeatedly entering user identification/application password combinations in attempt to gain unauthorized access to a restricted application.

Still another object of the invention is to provide an apparatus and method which can be adapted to include additional user and system information with the user identification and application password.

Accordingly, an aspect of the present invention is directed to a system for facilitating secondary access authentication to applications of a computer installation or host system. The disclosed system provides user information and an application password to an access system which grants a user access to an application where the user has been previously authenticated by a primary-level access authentication system. The system comprising an authorization code generator which creates an authentication code which may contain the user and system expiration information, and an authorization code interpreter which reads the authentication code and provides the user information and the other desired information and application password to the access system. The authentication code is partially validated by the authorization code interpreter in accordance with the user information, system information and expiration information prior to the authorization code interpreter providing the user information and the application password to the access system.

According to a feature of the present invention, the user information comprises at least one of a user identification, a system identification, a current date, a starting date, a starting time, and a valid duration period.

According to another feature, if the authorization code interpreter determines that the authorization code is not valid, the user information and the application password is not provided to the access system, and if the authorization code interpreter determines that the authorization code is valid, the user information and the application password is provided to the access system.

According to yet another feature, the user is prompted to re-enter the authorization code or enter a new application password if the authorization code is not valid.

According to still another feature, the authorization code interpreter comprises the access system and performs the secondary access authentication by comparing the user identification and the application password to a list of valid user identifications and application passwords.

According to another feature, the authentication code is encrypted by the authentication code generator. The authentication code may be encrypted by public key encryption methods, bit shuffling, byte or nibble swapping or other methods.

According to yet another feature, the authorization code interpreter decrypts the authentication code.

According to still another feature, the system provides the user information and the application password to a remote computer system.

According to another aspect of the present invention, a method of facilitating secondary access in a secondary access authentication system is provided in which the secondary access authentication system provides user information and an application password to an access system which grants a user access to an application. The method of secondary access authentication assumes that the user has been previously authenticated by a primary-level access authentication system. The method of facilitating secondary access comprises generating an authorization code, the authentication code containing the user information and an application password; reading the authentication code; interpreting the authorization code; validating the authorization code in accordance with the user information; and providing the user information and the application password to the access system.

According to a feature of the present invention, the user information comprises at least one of a user identification, a system identification, a current date, a starting date, a starting time, and a valid duration period.

According to another feature, the step of validating further comprises denying access to the application by not providing the user information and the application password to the access system if the authorization code is not valid.

According to yet another feature, the step of validating further comprises prompting the user re-enter the authorization code or enter a new application password if the authorization code is not valid.

According to still another feature, the method further comprises comparing the user identification and the application password to a list of valid user identifications and application passwords to grant the user access to the application.

According to another feature, the step of generating further comprises encrypting the authorization code.

According to yet another feature, the step of interpreting the authentication code further comprises decrypting the authorization code.

The above-listed and other objects, features and advantages of the present invention will be more fully set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is further described in the detailed description which follows, by reference to the noted plurality of drawings by way of non-limiting examples of preferred embodiments of the present invention, in which like reference numerals represent similar parts throughout the several views of the drawings, and wherein:

Fig. 1 illustrates an exemplary system architecture in which the present invention may be implemented;

Fig. 2 illustrates, in block diagram form, an exemplary system architecture in which the present invention may be implemented;

Fig. 3 is an exemplary flow chart of the processes and operations of the Authentication Code Generator of the present invention; and

Fig. 4 is an exemplary flow chart of the processes and operations of the Authentication Code Interpreter of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention provides an apparatus and method of secondary access authentication which is secure while providing users with the ability to publicly disclose an authentication code (AC) which is used to authenticate the user and gain access to an application. According to an aspect of the invention, embedded within the AC may be restrictions as to specific host systems, dates and times of access, expiration information, and a key to ensure that changes are made at specified intervals (e.g., by prompting the user for a new application password). Further information may be embedded into the AC as such information is deemed necessary. In accordance with another aspect of the present invention, a repository of application level password information is not necessary to track the expiration date and time of an application level password and/or application access (i.e., to perform aging). Further, in accordance with yet another aspect of the present invention, the AC may be used in conjunction with operating-system level authenticated user identification/password applications (i.e., those applications not requiring secondary access to run) without exposing the AC information.

The features and aspects of the present invention may be implemented by any suitable combination of hardware, software and/or firmware. By way of a non-limiting example, an exemplary computer installation in which the present invention may be implemented is illustrated in Fig. 1. As shown in Fig. 1, a local computer installation 1 may include a network server 10 and one or more personal computers (PCs) or workstations 20 that are connected as clients to the server via a network infrastructure 35. The network server 10 may comprise, for example, a UNIX-based or Windows NT-based computer platform having a single microprocessor or multiple processors (e.g., Intel Pentium Pro processor or Digital Equipment Company Alpha RISC processor), long-term storage (e.g., a RAID disk array having 10 Gb of storage), random access memory (RAM) (e.g., 128 Mb RAM), communication peripherals (e.g., network interface card, modem, and/or terminal adapter), and application programs (e.g., spreadsheet, wordprocessor and/or database related software applications) which may be distributed to clients residing on the network. The client PCs 20 may comprise Windows95-based personal computers having an Intel Pentium processor, long-term storage (e.g, a IDE or SCSI hard disk having 2 Gb of storage), random access memory (RAM) (e.g., 32 Mb RAM), communication peripherals (e.g., network interface card, modem, and/or terminal adapter), and application programs (e.g., spreadsheet, wordprocessor and/or database related software applications). The workstations 20 may comprise UNIX-based IBM RS/6000 or SUN SPARCSTATION workstations. The workstations 20 may have local hard-disks or may be configured as diskless clients to enhance network security. The network infrastructure 35 may comprise an Ethernet Local Area Network (LAN), a Transmission Control Protocol/Internet Protocol (TCP/IP) network, or Token Ring network. Also, one or more printers 15 (e.g., laser printers or plotters) may be connected to the network infrastructure 35 to provide for hard copy output.

One or more remote systems may be connected to the network infrastructure 35 via a network gateway (not shown) and communications links 40. In the exemplary computer installation 1, the remote systems may comprise a remote network server 25 (e.g., a network server similar to network server 10) or remote mainframe system 30 (e.g., an IBM AS/400) that are connected to the network infrastructure 35 by communications links 40. Although the communication links 40 to the remote systems are shown as direct connections, the communications links 40 may comprise a wireless network, satellite links, a Wide Area Network (WAN), dedicated lines, and a TCP/IP network. Users may access the remote server 25 or remote mainframe 30 using various protocols and client applications such as File Transfer Protocol (FTP), Telnet or UNIX rlogin.

Users may be authorized for primary access to the network infrastructure 35 by entering a user identification (user_id) and password at the one of the PCs or workstations 20. The user_id/password combination may be sent via the network infrastructure 35 to the host operating system residing at, for example, the network server 10 where it is compared to an access list, password file or database table, stored on, for example, the network server 10. If a valid user_id/password combination is entered, the user is granted access privileges by the host operating system in accordance with the user's predetermined user profile, which may also be stored on the network server 10. If the user_id/password combination is invalid, the user may be afforded a predetermined number of retries to enter a valid user_id/password combination, after which the PC or workstation 20 may be locked out from accessing the computer installation 1 via the network infrastructure 35. Once a user is granted access, the user may be presented with a graphical user interface (e.g., X-window interface) or a command prompt from which the user may launch and execute applications, perform file maintenance operations, interact with the remote systems, and communicate with other users (e.g., via E-mail).

When the user launches or executes an application, secondary access authentication techniques in accordance with the present invention may be utilized to authenticate users who are accessing particular applications. As noted above, the present invention is directed to a secondary access authentication mechanism by which the user's identity may be authenticated by the application or remote system by passing an authentication code (AC) to the application or system performing the secondary access authentication.

An exemplary implementation of the present invention is illustrated in Fig. 2. According to the embodiment of Fig. 2, the present invention may comprise a customized secondary access authentication system 45. In the exemplary implementation, the customized secondary access authentication system 45 may comprise an Authentication Code (AC) Generator 50 and an Authentication Code (AC) Interpreter 60. To briefly describe the customized secondary access authentication system 45 of the present invention, a user who desires access to an application or system requiring secondary access authentication first generates an authentication code (AC) using the AC Generator 50. The AC may contain information used by the application or system to authenticate a user's identity. Additional information may be contained within the AC as deemed necessary. Further, the AC may be encrypted to enhance security. The generated AC may be passed (via a file or user entry) as a character string to the AC Interpreter 60 in order to gain access to applications requiring secondary access authentication. The AC Interpreter 60 may verify the information contained with the AC and provide, for example, a user identification and application password to an access system 70 via an application programming interface (API) call. The access system 70 may be part of the application or system requiring secondary access authentication, or a stand-alone program, and determine if the user is authorized for access to the application or system. A more detailed discussion of the operations and functions of the AC Generator 50 and the AC Interpreter 60 follows below with reference to Figs. 3 and 4, respectively.

In the exemplary implementation of Fig. 2, the inputs to the AC Generator 50 may include the user's user identification (user_id) and application password, a system identification (system_id), and a current date (current_date). The application password may be entered by the user, whereas the user_id, current_date and system_id may be obtained using the appropriate operating system calls. The output of the AC Generator 50 is the authentication code (AC), which may be output as, for example, a character string in a file or to a display of the PC/workstation 20. As noted above, the AC may be encrypted in accordance with the security concerns of the computer installation 1.

The inputs to the AC Interpreter 60 may include the AC, which is input as a character string (e.g., by user entry or within a file) . If the AC is encrypted, the AC Interpreter 60 decrypts the AC using the complementary decryption technique to the encryption technique used by the AC Generator 50. Also input to the AC Interpreter 60 may be the user_id, system_id, and current_date, which are obtained operating system calls. These values may be compared to the user_id, system_id, and current_date contained within the AC to verify the user's identity. After the user's identity is verified, the user_id and application password may be passed to the access system 70 to determine if the user is authorized for access to the application software.

The AC Generator 50 and AC Interpreter 60 may be written in a high-level language (e.g., C or C++) or may be implemented by linking applets (e.g., Java), and may reside on the network server 10, PCs/workstations 20, remote server 25, or remote mainframe 30 as stand-alone programs or, modules incorporated and linked to other applications. Further, the AC Generator 50, AC Interpreter 60 and Access system 70 do not have to reside on the same network server 10, PC/workstation 20, remote server 25, or remote mainframe 30. In such an implementation, information (e.g., the AC, user_id and application password) may be passed between the AC Generator 50, AC Interpreter 60 and Access system 70 via API calls, scripts, and user prompted data entry.

Each PC or workstation 20 may be provided with an AC Generator 50 such that users may access the AC Generator 50 and generate ACs to be used to access to software applications. In an environment requiring enhanced security, access to the AC Generator 50 (as well as the source code of the AC Generator 50 and AC Interpreter 60) may be limited to, for example, a system administrator, who would generate ACs for users.

The AC Generator will now be described with reference to the exemplary flow chart shown in Fig. 3. A user desiring secondary access authentication for a particular application or system, executes the AC Generator program at step 100. The user may launch the AC Generator at the PC or workstation 20 via a GUI interface or by entering a command at an operating system command prompt. The AC Generator may be stored and executed on, e.g., the network server 10. The generated AC may be limited to the system, personal computer, or workstation on which it is generated, or may be passed to and used on other remote systems. For additional protection, the source code for the AC Generator may be restricted, such that only the executable code is accessible to users.

At step 102, the AC Generator identifies the user's user_id by making the appropriate operating system calls (e.g., UNIX whoami). Also at step 102, the AC Generator may use system calls to identify the system on which the user is logged-on (system_id), and the current date (current_date) if such information is desired to be incorporated into the AC. Further, the date may be an integer value calculated by the number of days, etc., from a base date (e.g., 1/1/97). At step 104, the AC Generator prompts the user to enter his or her application password (password). The user's application password that is entered at step 106 may be any allowable system password, and may be different than the user's primary-level password for added security. At step 108, the AC Generator creates a string containing the user_id, password, system_id and current_date by concatenating the information into a single character string. At step 110, the AC Generator encrypts the character string containing the user_id, password, system_id and current_date into an encrypted string of characters using, for example, public key encryption methods, bit shuffling, byte or nibble (i.e., half-byte) swapping, etc.

The encryption technique that is utilized at step 110 may be determined in accordance with a level of security required for the particular computer system and infrastructure. Further, additional or less information than that noted-above may be included in the character string prior to encryption. For example, a time period that the AC will be valid may be included, and an AC start date or time may also be included. As noted, the current_date field may also be converted into an offset from a base date if desired. The length of the resulting AC may vary in accordance with the amount of site-specific information that was included into the AC, or the encryption technique utilized.

At step 112, the AC Generator provides the encrypted AC string as output for subsequent use such that the user may gain access to applications and remote systems running the complementary AC Interpreter of the present invention (to be described below). The AC may be output as information contained in a file (e.g., an ASCII text file), sent to the printer 15, or output directly to the screen of the PC or workstation 20. At step 114, the functions and processes of the AC Generator end.

As a non-limiting example, a user may generate an AC as follows. Assume, for example, that a user "LTJ" on system "XYZ" having an application password of "QWERTY" desiring an AC that is valid until January 1, 1999 (which is calculated as an integer offset from an initial date) executes the AG Generator to create a starting string such as "LTJXYZQWERTY687". The character string may then be converted, for example, to hexadecimal format and nibble swapping may be performed in a set sequence to encrypt the starting string. Other encryption techniques may be used so long as the AC Interpreter utilizes a complementary decryption technique. The encrypted character string, which represents the generated AC may be, for example, "S!d5SH%LAAnET*&E#!". The generated AC may be provided to the user so that it may be written down, embedded into scripts, or entered directly by the user as it would only be valid and usable by the particular user on the particular system for a predetermined period of time, as described below.

Referring now to Fig. 4, the AC Interpreter of the present invention will now be described. According to an aspect of the present invention, an application to which the user desires access may utilize the AC Interpreter to read the AC and grant secondary access.

The AC Interpreter of the present invention may be used in conjunction with existing software applications requiring secondary access authentication. When the user launches a software application requiring secondary access authentication at the PC or workstation 20 (e.g., the GUI interface or a command entered at the command prompt), the AC Interpreter may be automatically executed as a linked module. The user may be prompted to enter the AC, the AC may be read from a file, or the AC may be passed to the AC Interpreter via a script or environmental variable. Alternatively, the AC Interpreter may be launched as a stand-alone program, and utilizing information contained within a validated AC, the AC Interpreter may pass the user's user_id and application password and launch the software application requiring secondary access authentication.

At step 200, the AC Interpreter reads the AC (e.g., "S!d5SH%LAAnET*&E#!") and at step 202, solicits the current_date, system_id and user_id from the operating system using the appropriate system calls. As noted above, the AC may be passed by the user to the AC Interpreter as a string embedded in a script, or entered directly by the user when accessing the application or remote system. The AC, the current_date, system_id and user_id values are temporarily stored by the AC Interpreter in a memory (not shown) of, for example, the network server 10. At step 204, the AC is decrypted in accordance with the encryption scheme utilized by the AC Generator to recreate the above-noted character string (e.g., "LTJXYZQWERTY687"). At step 206, the decrypted AC is then broken into its requisite component fields (e.g., user_id, password, system_id, and current_date) and the user_id, system_id and current_date are compared at step 208 to the temporarily stored values in memory solicited from the operating system at step 202. In order to break the AC into its requisite component fields, the AC Interpreter may include logic to specify how to break the AC into each of the component fields by predefining the field lengths of the user_id, password, system_id, and current_date. In the present example, the user_id has a field length of three characters, the password has field length of three characters, the system_id has a field length of six characters, and the current_date is an integer value. Other field lengths may be used in accordance with system requirements and preferences.

At step 210, the AC Interpreter determines if the AC is valid in accordance with the comparison performed at step 208. The AC may be determined to be invalid if any of the user_id, system_id, and current_date values do not match the values in memory. If the AC is invalid (e.g., expired, not for the present system, does not contain the user identification, or fails other pre-validation tests), the user may be prompted to obtain or update the application password and/or to reenter the AC, at step 214, where after the AC Interpreter terminates. If the AC is valid at step 210, then at step 212, the application password (and any other necessary information) is provided to the validation logic to perform the secondary access authentication in accordance with the application's specific requirements. For example, the validation logic may compare the user_id and application password with an application specific password repository to validate that the user is authorized to access the application.

Other application or system specific information may be included in the character string to provide additional security and authentication information. Further, the application specific information may be contained in a repository (located on, for example, server 10) and may include the user's application user_id and password. Further, if the AC includes date specific information, the AC Interpreter may determine if the AC is valid or has expired, if the user should be prompted to change his or her application password, or if the AC is not yet valid.

As the exemplary embodiment illustrates, the AC Interpreter of the present invention may be used in conjunction with existing software applications, including commercially available database applications such as Oracle Database Management System, available from Oracle Corp., of Redwood Shores, CA. The AC Interpreter may be linked to existing software applications as a module that is executed when the user accesses the application. In such an implementation, the module could validate the system, user and date information and pass the decrypted user_id and password in an application-recognized format to the application software for authentication. It is also possible to implement the AC Interpreter such that the AC Interpreter performs the secondary access authentication validation logic, rather than the application. In such an implementation, the AC Interpreter may authenticate the user and provide the application with an indication that the user is authorized to access the application code.

For new applications utilizing the AC directly, if the encryption and decryption mechanisms are contained within the application, security concerns may be reduced by conveying the operating system-level authentication to the application, because the operating system-level authentication is always required. When interfacing with application code that does not directly utilize the AC, disclosure of the encryption/decryption logic could result in exposure of the application user_id and password, thus resulting in a failure of security.

As is evident from the above, the apparatus and method of the present invention offers adequate levels of security for a variety of purposes, and by utilizing the appropriate restrictions to the AC Generator and AC Interpreter source code, a very secure and robust AC may be provided. Also, because the user information solicited by the application to perform secondary access authentication is dependent on a successful operating system-level authentication, the secondary level authentication via an AC remains secure, even though the AC may be made public.

In addition, as noted above, the AC may include additional data and information that the site deems necessary. For example, the AC may include client specific information such that the user may access the application from designated client PCs and workstations within the computer infrastructure. Also, the AC Generator and AC Interpreter may be provided with a mechanism to periodically (e.g., hourly or daily) change the encryption technique to further enhance security. Further, the AC Interpreter may operate as a stand-alone module or be incorporated into the application or system which requires secondary access authentication. Still further, the AC Generator and AC Interpreter may be utilized in an environment where the PC or workstation is connected to a network, but where the application requiring secondary access executes locally on the PC or workstation. Yet further, the AC Generator and AC Interpreter may be utilized in a stand-alone (i.e., non-networked) PC or workstation, which performs operating-system level authentication and is loaded with applications which perform secondary access authentication.

While the invention has been described with reference to several exemplary embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitations. Changes may be made, within the purview of the disclosure, as presently stated and as amended, without departing from the scope and spirit of the invention in its aspects.

WHAT IS CLAIMED:

1. A system for facilitating secondary access authentication, said system providing user information and an application password to an access system which grants a user access to an application, said user having been previously authenticated by a primary-level access authentication system, said system comprising:

an authorization code generator which creates an authentication code, said authentication code containing said user information and said application password; and

an authorization code interpreter which reads said authentication code and provides said user information and said application password to said access system,

said authentication code being validated by said authorization code interpreter in accordance with said user information prior to said authorization code interpreter providing said user information and said application password to said access system.

2. The system according to claim 1, wherein said user information comprises at least one of a user identification, a system identification, a current date, a starting date, a starting time, and a valid duration period.

3. The system according to claim 2, wherein if said authorization code interpreter determines that said authorization code is not valid, said user information and said application password is not provided to said access system, and wherein if said authorization code interpreter determines that said authorization code is valid, said user information and said application password is provided to said access system.

4. The system according to claim 3, wherein said user is prompted to reenter said authorization code if said authorization code interpreter determines that said authorization code is not valid.

5. The system according to claim 3, wherein said authorization code interpreter comprises said access system and performs said secondary access authentication by comparing said user identification and said application password to a list of valid user identifications and application passwords.

6. The system according to claim 2, wherein said authentication code is encrypted by said authentication code generator.

7. The system according to claim 6, wherein said authentication code is encrypted by public key encryption, bit shuffling, byte or nibble swapping.

8. The system according to claim 6, wherein said authorization code interpreter decrypts

said authentication code, and if said authorization code interpreter determines that said authorization code is not valid, said user information and said application password is not provided to said access system, and if said authorization code interpreter determines that said authorization code is valid, said user information and said application password is provided to said access system.

9. The system according to claim 8, wherein said user is prompted to reenter said authorization code if said authorization code interpreter determines that said authorization code is not valid.

10. The system according to claim 8, wherein said authorization code interpreter comprises said access system and performs said secondary access authentication by comparing said user identification and said application password to a list of valid user identifications and application passwords.

11. The system according to claim 2, wherein said system provides said user information and said application password to a remote computer system.

12. A system for facilitating secondary access authentication to applications of a computer-base system, said system providing user information and an application password to an access system which grants a user access to an application, said user having been previously authenticated by a primary-level access authentication system, said system comprising:

means for generating an authorization code, said authentication code containing said user information and said application password; and

means for interpreting said authorization code, said interpreting means reading said authentication code and providing said user information and said application password to said access system,

said authentication code being validated by said interpreting means in accordance with said user information prior to said interpreting means providing said user information and said application password to said access system.

13. The system according to claim 12, wherein said user information comprises at least one of a user identification, a system identification, a current date, a starting date, a starting time, and a valid duration period.

14. The system according to claim 13, wherein if said interpreting means determines that said authorization code is not valid, said user information and said application password is not provided to said accessing system, and wherein if said interpreting means determines that said

authorization code is valid, said user information and said application password is provided to said accessing system.

15. The system according to claim 14, wherein said user is prompted to reenter said authorization code if interpreting means determines said authorization code is invalid.

16. The system according to claim 14, wherein said interpreting means comprises said accessing system, said interpreting means comparing said user identification and said application password to a list of valid user identifications and application passwords to grant said user access to said application.

17. The system according to claim 13, wherein said authentication code is encrypted by said generating means.

18. The system according to claim 17, wherein said authentication code is encrypted by public key encryption, bit shuffling, byte or nibble swapping.

19. The system according to claim 17, wherein said interpreting means decrypts said authentication code, and if said interpreting means determines that said authorization code is not valid,, said user information and said application password is not provided to said accessing means, and if said interpreting means determines that said authorization code is valid, said user information and said application password is provided to said accessing means.

20. The system according to claim 19, wherein said user is prompted to reenter said authorization code if interpreting means determines that said authorization code is invalid.

21. The system according to claim 19, wherein said interpreting means comprises said accessing means, said interpreting means comparing said user identification and said application password to a list of valid user identifications and application passwords to grant said user access to said application.

22. The system according to claim 13, wherein said system provides said user information and said application password to a remote computer system.

23. A method of providing a secondary access authentication to applications in a host system, said method providing user information and an application password to an access system which grants a user access to an application, said user having been previously authenticated by a primary-level access authentication system, a method of facilitating secondary access comprising:

generating an authorization code, said authentication code containing said user information and said application password;

reading said authentication code;

interpreting said authorization code;

validating said authorization code in accordance with said user information; and

providing, after validating said authorization code, said user information and said application password to said access system.

24. The method according to claim 23, wherein said user information comprises at least one of a user identification, a system identification, a current date, a starting date, a starting time, and a valid duration period.

25. The method according to claim 24, wherein said validating further comprises denying access to said application by not providing said user information and said application password to said access system if said authorization code is not valid.

26. The method according to claim 25, wherein said validating further comprises prompting said user to reenter said authorization code if said authorization code is not valid.

27. The method according to claim 25, further comprising comparing said user identification and said application password to a list of valid user identifications and application passwords to grant said user access to said application.

28. The method according to claim 23, wherein said generating further comprises encrypting said authorization code.

29. The method according to claim 28, wherein said interpreting said authentication code further comprises decrypting said authorization code.

30. The method according to claim 29, wherein said encrypting and decrypting is performed in accordance with one of public key encryption, bit shuffling, byte and nibble swapping.

FIG. 1

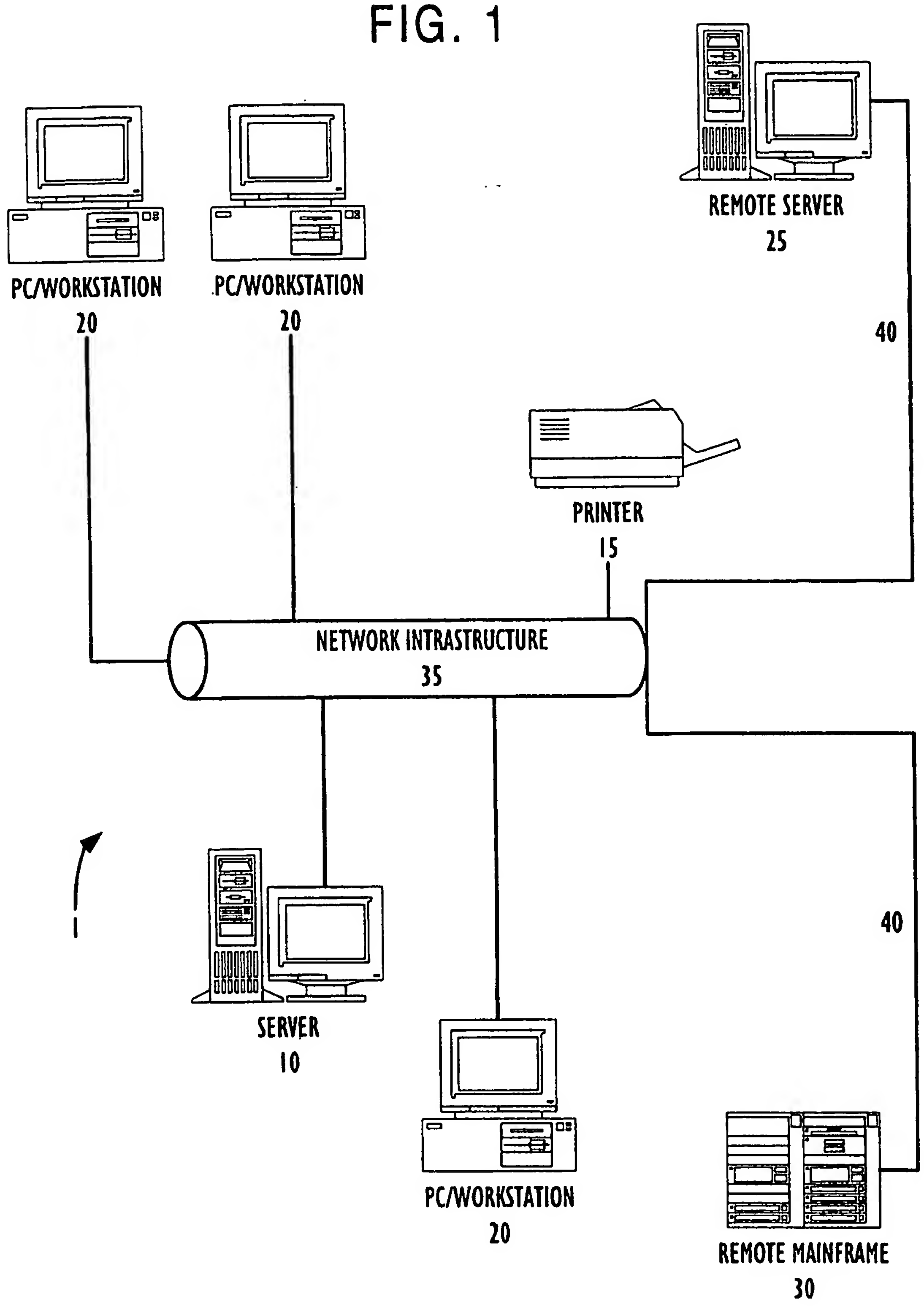

REMOTE SERVER
25
PC/WORKSTATION
20
PC/WORKSTATION
20
40
PRINTER
15
NETWORK INTRASTRUCTURE
35
1
SERVER
10
40
PC/WORKSTATION
20
REMOTE MAINFRAME
30

FIG. 2

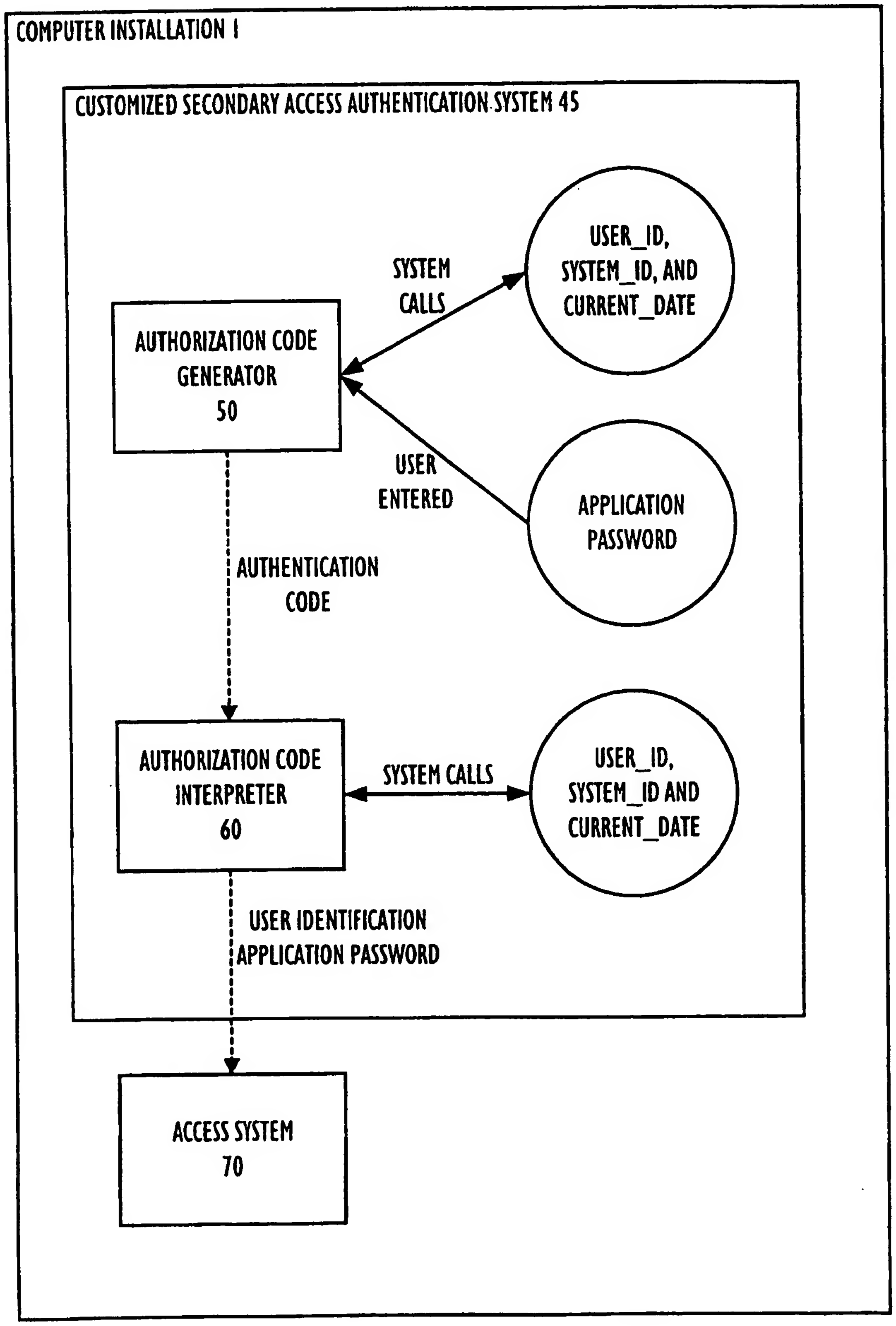

COMPUTER INSTALLATION 1
CUSTOMIZED SECONDARY ACCESS AUTHENTICATION SYSTEM 45
AUTHORIZATION CODE GENERATOR 50
SYSTEM CALLS
USER_ID, SYSTEM_ID, AND CURRENT_DATE
USER ENTERED
APPLICATION PASSWORD
AUTHENTICATION CODE
AUTHORIZATION CODE INTERPRETER 60
SYSTEM CALLS
USER_ID, SYSTEM_ID AND CURRENT_DATE
USER IDENTIFICATION
APPLICATION PASSWORD
ACCESS SYSTEM 70

FIG. 3

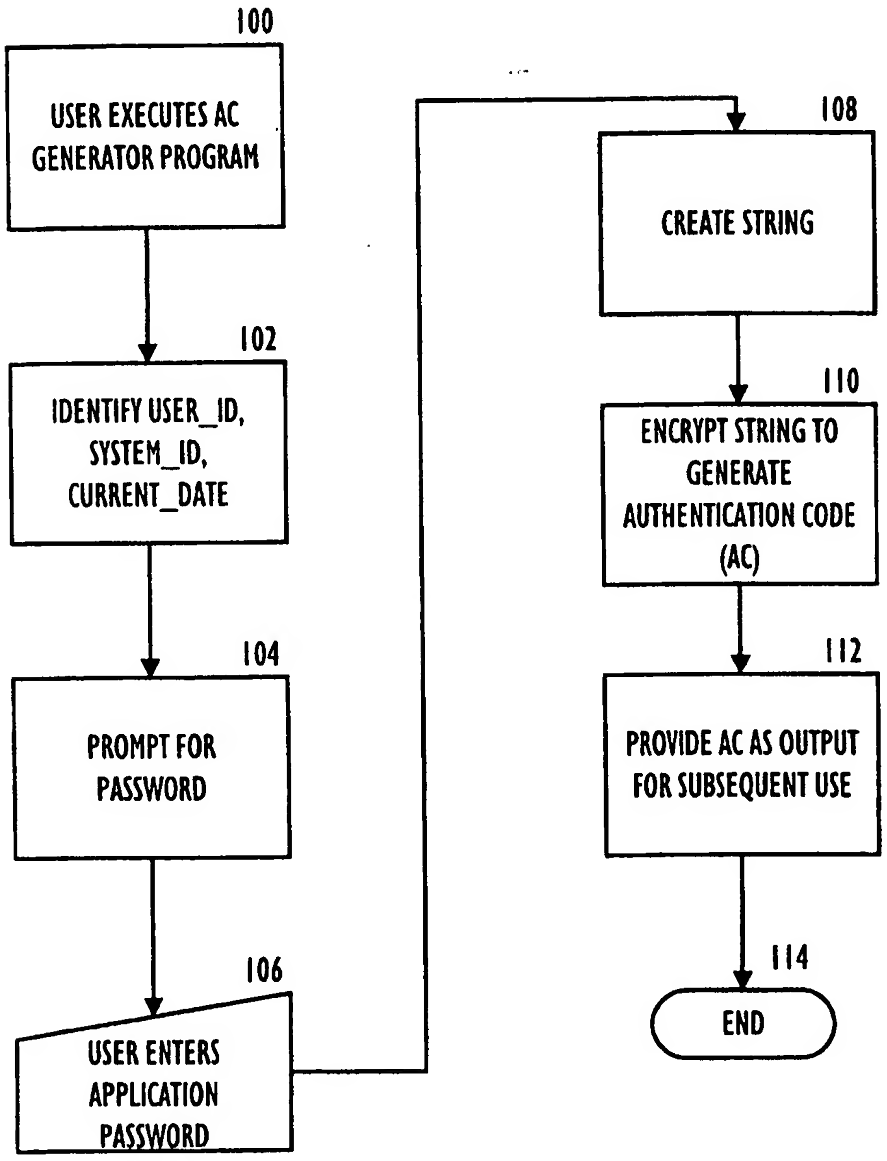
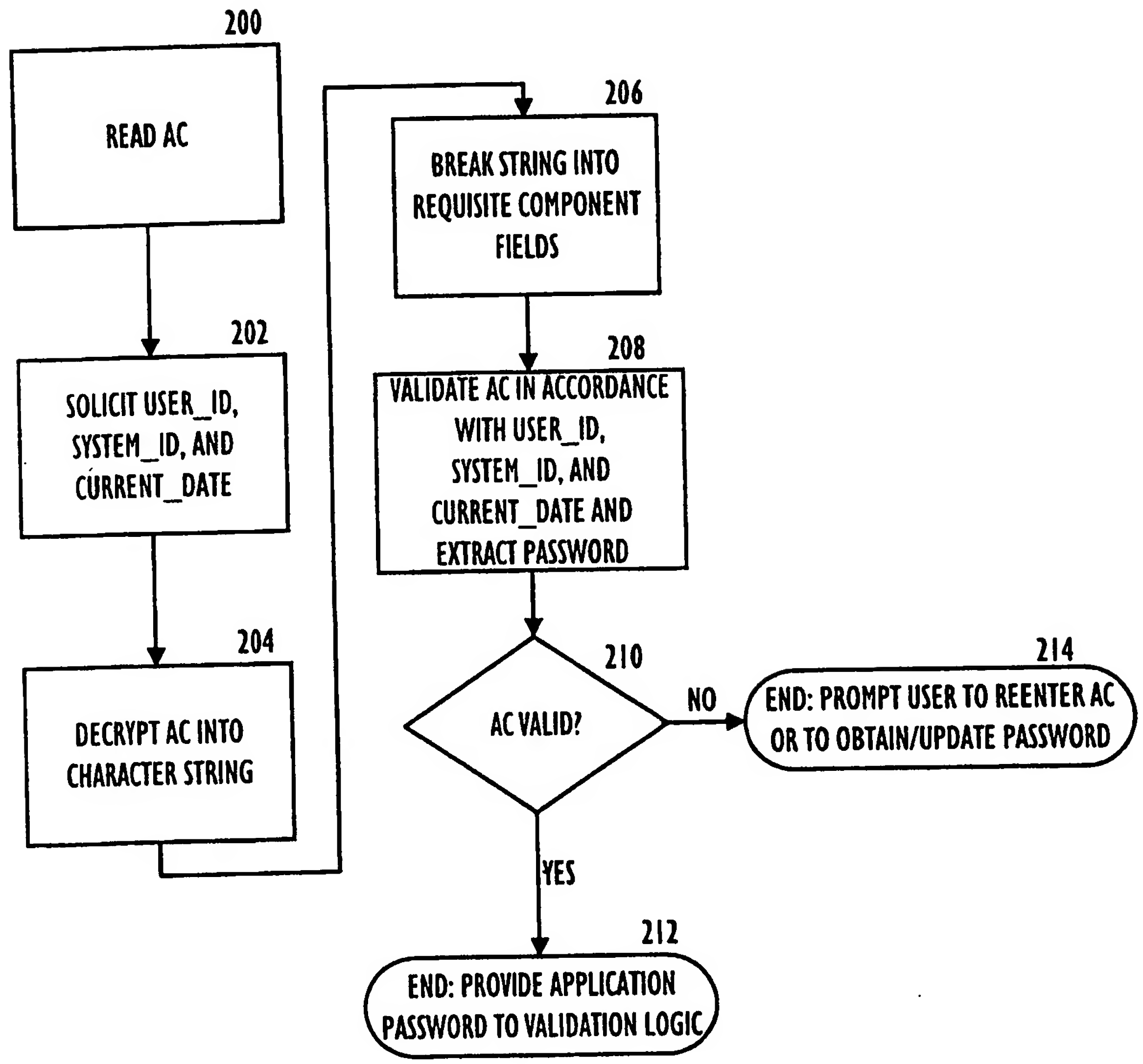

100
USER EXECUTES AC GENERATOR PROGRAM
102
IDENTIFY USER_ID, SYSTEM_ID, CURRENT_DATE
104
PROMPT FOR PASSWORD
106
USER ENTERS APPLICATION PASSWORD
108
CREATE STRING
110
ENCRYPT STRING TO GENERATE AUTHENTICATION CODE (AC)
112
PROVIDE AC AS OUTPUT FOR SUBSEQUENT USE
114
END

FIG. 4


200
READ AC
202
SOLICIT USER_ID, SYSTEM_ID, AND CURRENT_DATE
204
DECRYPT AC INTO CHARACTER STRING
206
BREAK STRING INTO REQUISITE COMPONENT FIELDS
208
VALIDATE AC IN ACCORDANCE WITH USER_ID, SYSTEM_ID, AND CURRENT_DATE AND EXTRACT PASSWORD
210
AC VALID?
NO
214
END: PROMPT USER TO REENTER AC OR TO OBTAIN/UPDATE PASSWORD
YES
212
END: PROVIDE APPLICATION PASSWORD TO VALIDATION LOGIC